ACEEE Int. J. on Information Technology , Vol. 3, No. 3, Sept 2013

# A Bitemporal SQL Extension

Georgia Garani
Department of Computer Science and Telecommunications, TEI of Larisa, Larisa, Greece
Email: garani@teilar.gr

***Abstract*—A bitemporal nested query language, BTN-SQL, is proposed in this paper. BTN-SQL attempts to fill some gaps present in currently available SQL standards. BTN-SQL extends the well-known SQL syntax into two directions, the user-friendliness support of nested relations and the effective support of bitemporal data. The schema of a bitemporal nested database is difficult to be understood since it is complicated by nature; therefore, an extended approach of the Entity-Relationship model, the BTN-ER model, is also proposed for modelling complex bitemporal nested data.**

***Index Terms*—bitemporal data model, nested relation, temporal attribute, temporal element, SQL, ER model**

## I. Introduction

Time is considered to be the fourth dimension, additionally to the three dimensions of space. Changes that are constantly happening in the real world are expressed through time which keeps track of these changes.

It is difficult to record time because it consists of many granularities. Time can depict iteration, periodicity and divergence, as well as duration.

Databases must be capable of modelling and manipulating temporal data along with all other data. These databases are called temporal databases. In the database world, two different aspects of time are of interest, the time that an event takes place in the real world named as valid time and the time that the event is stored in the database named as transaction time. The distinction of these two time domains can be illustrated by two simple questions, "When was John promoted to a manager?" which gives historical information (valid time) and "When does the database believe that John was promoted to a manager?" which gives rollback information (transaction time). The two questions above may give different answers. Relations are called tuple timestamping when whole tuples are timestamped, or attribute timestamping, when individual time-varying attributes are timestamped.

A relation where tuples have the same lifespan in each attribute of a tuple is called homogeneous; otherwise, it is called heterogeneous. By definition, a homogeneous relation is a special case of a heterogeneous relation. Tuple timestamping models are homogeneous because of their structure. In contrast, attribute timestamping models can be either homogeneous or heterogeneous.

Time in temporal databases is represented by one of the three different approaches, a single time point, two time points which represent a time interval or a set of time intervals which form a temporal element. A time point indicates either the start or the end of the lifespan of an object (relation, tuple or attribute). However, the whole history of the object is stored using two different attributes, i.e. the start point and the stop point. Time intervals contain the complete information about the lifespan of an object compacted. Temporal elements are defined as the union of disjoint and non-adjacent time intervals.

SQL, one of the first commercial languages for the relational model, became a standard of the American National Standards Institute (ANSI) in 1986 and of the International Organization for Standards (ISO) in 1987. Since then, the standard has been improved with added features. Nowadays, SQL:2011 is the latest revision of SQL formally adopted in December 2011. One of the most important feature in SQL:2011 is the ability to create and manipulate temporal tables [1].

However, there is a gap between theory (relational model) and implementation (SQL), despite the many promising solutions proposed by researchers over the years. Only recently have commercial DBMSs started to support temporal features in SQL. Before that, the handling of time in SQL was time-consuming, error-prone and even sometimes impossible.

In this paper, an extension of SQL is proposed which extends temporal and nested features of SQL. The SQL extension, called BTN-SQL, is based on the Bitemporal Nested Model (BTNM) presented in [2]. The proposed extension is fully compatible with SQL standard.

The rest of the paper is organized as follows. Section II presents terminology which will be used in subsequent sections. In Section III the structure of BTN relations is explained. Section IV presents briefly BTN algebra. In Section V an extended version of the ER model is presented. A case study is given in Section VI. BTN-SQL extension is presented in Section VII, related work is discussed in Section VIII and finally, last section concludes the paper.

## II. Preliminaries

Time in BTNM is represented by temporal elements [2].

*Definition 1 (Temporal element, TE)* Temporal element is a finite set of disjoint and non-adjacent time intervals.

*Example 1.* TE = {[1, 5), [8, 9), [15, 23)}.

*Definition 2 (Temporal element's start point, START)* The start point of a temporal element is the minimum start point of all the start points of the time intervals that belong to this temporal element.

TE = {[$t_{1START}$, $t_{1STOP}$), [$t_{2START}$, $t_{2STOP}$), …, [$t_{kSTART}$, $t_{kSTOP}$)}, where $k \geq 1$.

START(TE) = MIN{$t_{1START}$, $t_{2START}$, …, $t_{kSTART}$}

*Definition 3 (Temporal element's stop point, STOP)* The stop point of a temporal element is the maximum stop point of all the stop points of the time intervals that belong to this temporal element.

STOP(TE) = MAX{$t_{1STOP}$, $t_{2STOP}$, …, $t_{kSTOP}$}


DOI: 01.IJIT.3.3.1249

Temporal elements are closed under the set theoretic operations of union, difference and intersection, which are defined next.

Let $TE_1$ and $TE_2$ be two temporal elements. Then, the following definitions are given:

*Definition 4 (Union of temporal elements, $\cup_{TE}$ )*
It is the temporal element defined as

$$TE_1 \cup_{TE} TE_2 = \{ t \mid t \in TE_1 \vee t \in TE_2 \}.$$

*Definition 5 (Difference of temporal elements, $-_{TE}$)*
It is the temporal element defined as

$$TE_1 -_{TE} TE_2 = \{ t \mid t \in TE_1 \wedge t \notin TE_2 \}.$$

*Definition 6 (Intersection of temporal elements, $\cap_{TE}$ )*
It is the temporal element defined as

$$TE_1 \cap_{TE} TE_2 = = \{ t \mid t \in TE_1 \wedge t \in TE_2 \}.$$

*Definition 7 (Temporal attribute, $R_t$)* A *temporal attribute* is an attribute defined on the domain of temporal elements.

Temporal elements which timestamp attribute values, either valid time or transaction time, form temporal attributes. Each atomic attribute which changes over time has two temporal attributes connected with it, a valid temporal attribute and a transaction temporal attribute.

*Definition 8 (Valid temporal attribute, $R_{Vt}$)* A valid temporal attribute is a temporal attribute which shows for each tuple the time period over which each value of the atomic attribute is valid (valid time).

*Definition 9 (Transaction temporal attribute, $R_{Tt}$)* A transaction temporal attribute is a temporal attribute which shows for each tuple the time period over which each value of the atomic attribute is stored in the database (transaction time).

Temporal attributes, either valid or transaction, in the same relation can be defined over different time domains.

*Definition 10 (Bitemporal nested attribute, $R_{btn}$)* The atomic attribute and the corresponding temporal attributes (valid temporal attribute and transaction temporal attribute), referred to on the whole as a *bitemporal nested attribute*.

Bitemporal nested attributes form bitemporal nested subrelations in the general case. However, bitemporal attributes may also appear at the top level of relations. Therefore, time-varying attributes are timestamped by taking advantage of the nested feature of the model.

An attribute path specifies the nesting level of an attribute, its name and the nesting path to reach it.

*Definition 11 (Attribute path, P)*

Let $P_{A_{btn} \to A_j}$ be the path of a bitemporal nested or atomic attribute $A_j$ belonging to bitemporal nested attribute $A_{btn}$, which is a child of the root of relation R. Then, $P_{A_{btn} \to A_j}$ is defined as follows:

i) $P_{A_{btn} \to A_j} = A_{btn}$, where $A_j = A_{btn}$

ii) $P_{A_{btn} \to A_j} = A_{btn}(L_{A_{btn+1} \to A_j})$, where $A_{btn+1}$ is an attribute of $A_{btn}$ either equal to or containing $A_j$.

Then, the set of all attributes (atomic and bitemporal nested) of R can be defined as

$$Attr(R) = \{R_{a1}, R_{a2}, \ldots, R_{ap}, R_{btn1}, \ldots, R_{btni}, \ldots, R_{btnq}\} =$$

$$\{R_{a1}, R_{a2}, \ldots, R_{ap}, R_{btn1}, \ldots, \bigcup_{k=0}^{m} P_{R_{btni} \to R_{btni}{}_k}, \ldots, R_{btnq}\}$$

where $R_{a1}, R_{a2}, \ldots, R_{ap}$ are atomic attributes at nesting level 1 of relation R ($p \geq 0$), $R_{btn1}, \ldots, R_{btni}, \ldots, R_{btnq}$ are bitemporal nested attributes at nesting level 1 of relation R ($1 \leq i \leq q$), m is the number of descendants' attributes of nested attribute $R_{btni}$ and

$$R_{btni}{}_k = \begin{cases} R_{btni} \text{ for } k = 0 \\ R_{btni}{}_k \text{ for } k \neq 0 \text{ (i.e. an attribute that has temporal nested attribute } R_{btni} \text{ as its ancestor)} \end{cases}$$

*Definition 12 (Join path, L)* If R is a relation scheme and $A_i$, $A_j$ are schemes of subrelations of R, then a join path, $L_{A_i \to A_j}$, from $A_i$ to $A_j$, is recursively defined as follows.

(i) $L_{A_i \to A_j} = \varnothing$ if $A_i = A_j$,

(ii) $L_{A_i \to A_j} = Ai(Ai_{+1} L_{A_{i+1} \to A_j})$, where $L_{A_{i+1} \to A_j}$ is a join path from $A_i$ to $A_j$. [3]

## III. BTNM RELATIONS

A relation in the BTNM is a bitemporal nested relation consisting of atomic, temporal (valid and transaction), nested (non-bitemporal) and bitemporal nested attributes.

The schema of a bitemporal nested relation r is denoted as $R=\{Attr(R_a), Attr(R_t), Attr(R_n), Attr(R_{btn})\}$ (Fig. 1) where $Attr(R_a) = \{R_{a1}, \ldots, R_{ak}\}$ the subset of

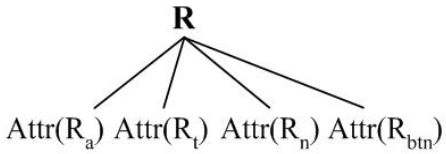


Figure 1. The general schema of a bitemporal nested relation r in the BTNM

all the atomic attributes of R, $Attr(R_t) = \{R_{Vt}, R_{It}\}$ valid temporal attribute and transaction temporal attribute of R, $Attr(R_n) = \{R_{n1}, \ldots, R_{nm}\}$ the subset of all the nested attributes of R where $R_{n1} = \{R_{n1}{}^1, R_{n1}{}^2, \ldots\}$, $R_{n2} = \{R_{n2}{}^1, R_{n2}{}^2, \ldots\}$ etc., $Attr(R_{btn}) = \{R_{btn1}, \ldots, R_{btnm'}\}$ the subset of all the bitemporal nested attributes of R where $R_{btn1} = \{R_{btn1}{}^1, R_{btn1}{}^2, \ldots\}$, $R_{btn2} = \{R_{btn2}{}^1, R_{btn2}{}^2, \ldots\}$ etc. and k, m, and m′ are positive integers.

The tree representation of R is shown in Fig. 2. Specifically, a relation r in the BTNM can be described as a tree with root node R and with all the nested and bitemporal nested attributes, $R_n$ and $R_{btn}$ respectively, as non-leaf nodes of the tree and all the atomic and temporal attributes, $Attr(R_a)$, $R_{Vt}$ and $R_{Tt}$ respectively, as leaves of the tree.

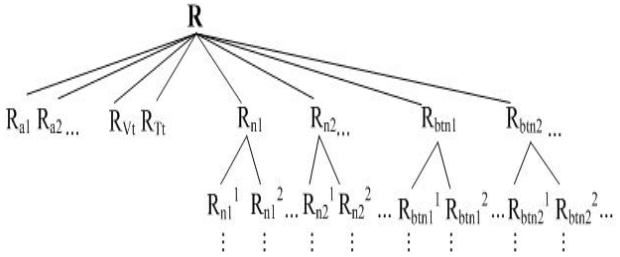


Figure 2. The tree representation of bitemporal nested relation r schema


DOI: 01.IJIT.3.3.1249

The proposed model is neither a *pure* tuple timestamping model nor a *pure* attribute timestamping model. However, the advantages of tuple timestamping and attribute timestamping models are combined and the limitations of both approaches are minimized since the temporal dimension of the model is nested and is not integral with the corresponding time-dependent value as in other previous proposed temporal nested models, e.g. [4]. As a result, the full power of the nested model is gained and simultaneously temporal elements can be readily referenced with or without their associated time-varying attribute values.

A BTNM relation may have a compound key. Key attributes can be time-varying when it is semantically appropriate, therefore a single valid timestamp and a single transaction timestamp is applied to the whole key whether a simple nested attribute or a compound key providing thus, the lifespan for the whole tuple. In addition, a bitemporal nested attribute can be a key attribute. A bitemporal nested key is a group of the relation's bitemporal nested attributes satisfying the constraint that two entities for which the sets composed of groups of values for the key attributes cannot be identical [3].

The representation of BTNM relations have many advantages compared to previously defined bitemporal models where the time domain of an attribute value is part of the same attribute as that value. Firstly, data describing an object is not fragmented into many relations since they can be nested within the same relation. Moreover, extra operations such as Temporal Atom Decomposition, Temporal Atom Formation and Drop-Time [5] are avoided. In addition, when the relation is viewed from the external level it can be characterised as an attribute timestamping relation, while at an internal level (that of a temporal subrelation) it can be viewed as a tuple timestamping relation. Therefore, atomic values and time values form different attributes and so can be referenced separately, an important feature since they have different properties.

## IV. BTN-ER Diagram

Entity-Relationship (ER) formalism is used as a database design tool for data modelling. However, there is no effective formalism similar to ER for modelling complex nested data apart from the traditional approach where doubled ellipse is used for modelling multivalued properties. For this purpose, an extended version of the ER model is presented in this section, called BTN-ER.

BTN-ER shows, in a simply and comprehensively way, a database schema from the conceptual point of view. Each entity is presented by a rectangle divided into three parts in the following order, the name of the relation, the primary key and lastly, the remaining attributes. An attribute with the prefix '*' denotes that this is a nested attribute, i.e. a subrelation. Attributes belonging to a nested attribute are included in a nested rectangle immediately below their parent nested attribute. The primary key can be atomic or nested following similar modelling.

BTN-ER model is demonstrated in Fig. 3. Specifically, relation A consists of three attributes, attribute_A1 which is the primary key of A, attribute_A2, an atomic attribute of A and attribute_A3, a nested attribute of A consisting of two atomic attributes, attribute_A3.1 and attribute_A3.2. Relation B consists of four attributes, one nested attribute, attribute_B1 which is the primary key of B, consisting of two atomic attributes, attribute_B1.1 and attribute_B1.2, atomic attributes, attribute_B2 and attribute_B3 and finally, nested attribute_B4 consisting of one atomic attribute, attribute_B4.1.

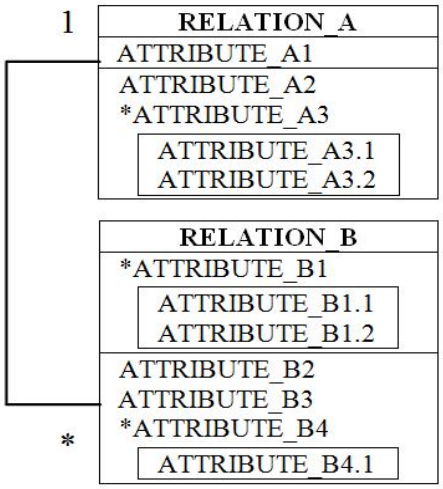


Figure 3. BTN-ER modelling

## V. The BTNM Algebra

In this section, a brief presentation of the operations of the BTNM algebra [2] is given. All the operations of are defined recursively. In the general case, temporal attributes are connected to the corresponding time-varying attributes to represent bitemporal data. Hence, a bitemporal nested attribute is formed by a time-varying attribute together with the corresponding valid temporal attribute and the transaction temporal attribute. In Table I all the operations presented in BTN algebra are listed together with their symbolism. The subscript '$_{bt}$' denotes the bitemporal version of each operation.

## VI. Case Study

The IT_COMPANY database consists of three relations, BiT_LOCATION, BiT_COURSE and BiT_TRAINING. The BTN-ER diagram of IT_COMPANY database is given in Fig. 4.

Relation BiT_LOCATION (Table II) contains data about branches of different companies. It consists of one atomic attribute, COMPANY and one bitemporal nested attribute, ANNEX. Subrelation ANNEX consists of two atomic attributes, BUILDING and ADDRESS and two temporal attributes, $ADDRESS\_PER_{Vt}$ and $ADDRESS\_PER_{Tt}$. The attributes of relation LOCATION have the following semantics: COMPANY – company name, BUILDING – building name, ADDRESS – street name, $ADDRESS\_PER_{Vt}$ –


DOI: 01.IJIT.3.3.1249

TABLE I. BTNM ALGEBRAIC OPERATIONS

| OPERATION | NON-RECURSIVE | RECURSIVE |
|---|---|---|
| UNION | $\mathbf{r \cup_{bt} q}$<br>Non-recursive union for bitemporal flat relations r, q | $\mathbf{r \cup_{bt}^{\cup} q}$<br>Recursive union for bitemporal nested relations r, q |
| DIFFERENCE | $\mathbf{r -_{bt} q}$<br>Non-recursive difference for bitemporal flat relations r, q | $\mathbf{r -_{bt}^{-} q}$<br>Recursive difference for bitemporal nested relations r, q |
| INTERSECTION | $\mathbf{r \cap_{bt} q}$<br>Non-recursive intersection for bitemporal flat relations r, q | $\mathbf{r \cap_{bt}^{\cap} q}$<br>Recursive intersection for bitemporal nested relations r, q |
| PROJECTION | $\mathbf{\pi_{bt}(r)}$<br>Projection of the whole bitemporal nested relation r<br><br>$\mathbf{\pi_{bt}(r(R_{btn}))}$<br>Non-recursive projection of a bitemporal nested attribute $R_{btn}$ at the top level of R | $\mathbf{\pi_{bt}^{\pi}(rL_{\pi})}$<br>Recursive projection for bitemporal nested relation r with $L_{\pi}$ a project path of r |
| BTIMESLICE | $\mathbf{s_{TE}(t[R_{btn}])}$<br>Non-recursive timeslice for a bitemporal nested attribute $R_{btn}$ of a bitemporal nested relation r along a given temporal element TE | $\mathbf{s^{s}_{TE}(r)}$<br>Recursive timeslice for bitemporal nested relation r along a given temporal element TE |
| SELECTION | $\mathbf{\sigma_{bt}(r_{c_{bt}})}$<br>Non-recursive selection concerning the set of temporal attributes Attr($R_t$) at the top level of a bitemporal nested relation r with $c_{bt}$ a set of conditions which must be true for the subset Attr($R_t$) | $\mathbf{\sigma_{bt}^{\sigma}(r_{c_{bt}}L_{\sigma})}$<br>Recursive selection for bitemporal nested relation r with $c_{btn}$ be a set of conditions in r and $L_{\sigma}$ a select list of r |
| UNNEST | $\mathbf{\mu_{bt}(r(R_{btn}))}$<br>Non-recursive unnest of a bitemporal nested attribute $R_{btn}$ at the top level of r | $\mathbf{\mu_{bt}^{\mu}(r(R_{btn}L_{btn}))}$<br>Recursive unnest of a bitemporal nested attribute $R_{btn}$ at the top level of r where $L_{btn}$ is an unnest list of the nested attribute $R_{btn}$ |
| NEST | $\mathbf{\nu_{bt}(r_{A_{btn}\rightarrow A})}$<br>Non-recursive nest of $A_{btn}$, the set of attributes at the top level of r, to form a new nested attribute A | $\mathbf{\nu_{bt}^{\nu}(r_{A_{btni}L_{btni}\rightarrow A})}$<br>Recursive nest of a set of attributes belonging in bitemporal nested attribute $A_{btni}$, at the top level of r, that are going to be nested to form a new nested attribute A, with $L_{btni}$ a nest list of $A_{btni}$ |
| RENAME | $\mathbf{\rho[Ri \leftarrow Ri']}(\mathbf{R})$<br>Non-recursive rename of an atomic or bitemporal nested attribute Ri to Ri′ at the top level of relation r | $\mathbf{\rho_{bt}^{\rho}[A_i \leftarrow A_i']}(\mathbf{R})$<br>Recursive rename of an atomic or bitemporal nested attribute A i to Ai′ at a lower level of relation r |
| CARTESIAN PRODUCT | $\mathbf{\times_{bt}(r, q)}$<br>Non-recursive Cartesian product between two bitemporal nested relations r and q with L and M, join paths of r and q respectively, empty | $\mathbf{\times_{bt}^{\times}(rL, Mq)}$<br>Recursive Cartesian product between two bitemporal nested relations r and q with L and M, join paths of r and q respectively, not empty |
| JOIN | $\mathbf{\triangleright\triangleleft_{bt}(r, q)}$<br>Non-recursive join between two bitemporal nested relations r and q with L and M, join paths of r and q respectively, empty | $\mathbf{\triangleright\triangleleft_{bt}^{\bowtie}(rL, qM)}$<br>Recursive join between two bitemporal nested relations r and q with L and M, join paths of r and q respectively, not empty |

time during which a company's annex was at a specific address and ADDRESS_PER$_{Tt}$ – time during which a company's annex at a specific address was stored in the database.

Relation BiT_COURSE (Table III) has data about different courses. It consists of four attributes, bitemporal nested attribute COURSE, atomic attributes COURSE_DURATION and TITLE and nested attribute SUBJECT. Subrelation COURSE consists of one atomic attribute, CN, and two temporal attributes, CN_PER$_{Vt}$ and CN_PER$_{Tt}$. The meaning of the attributes of relation BiT_COURSE is: CN - course number, CN_PER$_{Vt}$ – time duration of each course, CN_PER$_{Tt}$ – time duration of each course as it is stored in the database,


DOI: 01.IJIT.3.3.1249

COURSE_DURATION – course duration (number of hours), TITLE – course title and TOPICS – course topics.

Relation BiT_TRAINING (Table IV) contains data about courses and trainers provided by IT companies. It consists of one atomic attribute, COMPANY, and one bitemporal nested attribute, TRAINER. Subrelation TRAINER consists of one atomic attribute, TRN and one bitemporal nested attribute, COURSE. Subrelation COURSE consists of one atomic attribute, CN, and two temporal attributes, $CN\_PER_{Vt}$ and $CN\_PER_{Tt}$. Semantically, the attributes of the BiT_TRAINING relation have the following meaning: COMPANY – company name, TRN – trainer name, CN – course number (a course consists of a number of different topics), $CN\_PER_{Vt}$ – time duration of each course, $CN\_PER_{Tt}$ – time duration of each course as it is stored in the database.

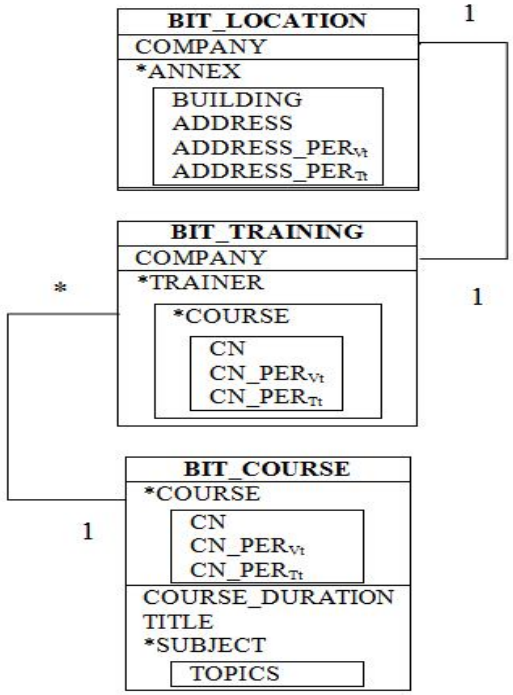


Figure 4. BTN-ER diagram of IT_COMPANY database

TABLE II. BIT_LOCATION RELATION

| COMPANY | ANNEX | | | |
|---|---|---|---|---|
| | BUILDING | ADDRESS | $ADDRESS\_PER_{Vt}$ | $ADDRESS\_PER_{Tt}$ |
| Toshiba | North Building | Porchester Rd. | [3/8/2005, 1/1/2010) | [10/10/2006, now) |
| IBM | Maple House | Kendal Av. | [17/1/2006, 22/5/2008) | [17/2/2006, 22/5/2008) |
| | Main Building | Danebury Rd. | [10/6/2008, 1/1/2010) | [10/10/2009, now) |
| Microsoft | Pegasus House | Ashford St. | [29/10/2004, 4/4/2007) | [29/10/2004, 1/4/2007) |
| | Queen's Building | Park Rd. | [18/3/2005, 1/1/2010) | [18/3/2005, now) |

*Query 1 (BTNM algebra).* List all companies and the courses they offered.

$\pi_t^{\pi}$(BiT_TRAINING(COMPANY, TRAINER(COURSE)))

The result of query 1 is shown in Table V.

*Query 2 (BTNM algebra).* List all trainers and the titles of the courses they have taught.

TABLE III. BIT_COURSE RELATION

| COURSE | | | COURSE DURATION | TITLE | SUBJECT |
|---|---|---|---|---|---|
| CN | $CN\_PER_{Vt}$ | $CN\_PER_{Tt}$ | | | TOPICS |
| 5.0 | [27/8/2005, | [27/8/2005, | 35 | Presentation Skills | Power Point<br>Word<br>Outlook Express |
| 3.3 | [17/1/2007, 28/4/2007) | [18/1/2007, 10/12/2010 | 15 | Multimedia | Power Point<br>Internet |
| 3.5 | [17/8/2007, 10/1/2010) | [1/9/2007, 10/1/2010) | 180 | Computer Skills | Access<br>Excel |
| 5.4 | [1/1/2005, 6/3/2005) | [1/1/2005, 6/3/2005) | | | |
| 5.2 | [13/2/2004, 4/3/2005) | [1/1/2005, now) | 80 | Programming | C++<br>JAVA |

TABLE IV. BIT_TRAINING RELATION

| COMPANY | TRAINER | | | |
|---|---|---|---|---|
| | TRN | COURSE | | |
| | | CN | $CN\_PER_{Vt}$ | $CN\_PER_{Tt}$ |
| Apple | Jack | 5.2 | [2/11/2004, 25/4/2005) ∪ [7/8/2006, 1/1/2010) | [16/3/2008, now) |
| | Mark | 3.3 | [2/1/2002, 8/11/2006) | [2/1/2002, 8/11/2006) |
| | | 3.5 | [30/4/2005, 1/1/2010) | [1/1/2006, now) |
| IBM | Tim | 5.2 | [19/3/2007, 21/4/2007) | [20/3/2007, 25/4/2007) |
| | | 5.0 | [17/12/2005, 1/1/2010) | 17/12/2005, now) |
| Microsoft | Karen | 3.3 | [25/6/2006, 1/1/2010) | [25/6/2006, now) |

$\pi_{bt}^{\pi}$((BiT_TRAINING$\triangleright\triangleleft_{bt}$BiT_COURSE)TRN, TITLE)

The result of query 2 is shown in Table VI.

TABLE V. RESULT OF QUERY 1

| COMPANY | TRAINER | | |
|---|---|---|---|
| | COURSE | | |
| | CN | $CN\_PER_{Vt}$ | $CN\_PER_{Tt}$ |
| Apple | 5.2 | [2/11/2004, 25/4/2005) ∪ [7/8/2006, 1/1/2010) | [16/3/2008, now) |
| | 3.3 | [2/1/2002, 8/11/2006) | [2/1/2002, 8/11/2006) |
| | 3.5 | [30/4/2005, 1/1/2010) | [1/1/2006, now) |
| IBM | 5.2 | [19/3/2007, 21/4/2007) | [20/3/2007, 25/4/2007) |
| | 5.0 | [17/12/2005, 1/1/2010) | [17/12/2005, now) |
| Microsoft | 3.3 | [25/6/2006, 1/1/2010) | [25/6/2006, now) |

TABLE VI. RESULT OF QUERY 2

| TRN | TITLE |
|---|---|
| Tim | Presentation Skills |
| Karen | Multimedia |
| Mark | Computer Skills |
| Jack | Programming |

*Query 3 (BTNM algebra).* Find all information for trainers Mark and Tim and for courses that took place for period overlapping the time interval [1/1/2007, 1/1/2008).

$\sigma_{bt}^{\sigma}$(BiT_TRAINING$_{(TRAINER(TRN)='Mark' \text{ OR } 'Tim') \text{ AND } (TRAINER(COURSE(CN_PERvt)) \textit{ overlaps } [1/1/2007,\ 1/1/2008))}$)

The result of query 3 is shown in Table VII.


DOI: 01.IJIT.3.3.1249

TABLE VII. RESULT OF QUERY 3

| COMPANY | TRAINER | | | |
|---|---|---|---|---|
| | TRN | COURSE | | |
| | | CN | $CN\_PER_{Vt}$ | $CN\_PER_{Tt}$ |
| Apple | Mark | 3.5 | [30/4/2005, 1/1/2010) | [1/1/2006, now) |
| IBM | Tim | 5.2 | [19/3/2007, 21/4/2007) | [20/3/2007, 25/4/2007) |
| | | 5.0 | [17/12/2005, 1/1/2010) | [17/12/2005, now) |

*Query 4 (BTNM algebra).* List the starting time point of every course that each trainer has given.

$\pi_{bt}^{\pi}$(BiT_TRAINING(TRN, CN, $\rho_{bt}^{\rho}$[*start*($CN\_PER_{vt}$) ←FIRST_DAY]))

The result of query 4 is shown in Table VIII.

TABLE VIII. RESULT OF QUERY 4

| TRN | (CN FIRST_DAY) | |
|---|---|---|
| | CN | FIRST_DAY |
| Jack | 5.2 | 2/11/2004 |
| Mark | 3.3 | 2/1/2002 |
| | 3.5 | 30/4/2005 |
| Tim | 5.2 | 19/3/2007 |
| | 5.0 | 17/12/2005 |
| Karen | 3.3 | 25/6/2006 |

*Query 5.* Find the titles of courses that were taught by Mark as known by the database system at 31/12/2009.

$\pi_{bt}^{\pi}$(($\sigma_{bt}^{\sigma}$(BiT_TRAINING $\triangleright\triangleleft_{bt}^{\triangleright\triangleleft}$ BiT_COURSE)

(CN_PERTt *contains* [31/12/2009, 31/12/2009) AND TRN = 'Mark')) (TITLE))

The result of query 5 is shown in Table IX.

TABLE IX. RESULT OF QUERY 5

| TITLE |
|---|
| Computer Skills |

*Query 6 (BTNM algebra).* Find the courses that have been given by trainers who work for IBM and had completed before IBM moved from Maple House.

$\pi_{t}^{\pi}$(($\sigma_{t}^{\sigma}$(BiT_TRAINING $\triangleright\triangleleft_{t}^{\triangleright\triangleleft}$ BiT_LOCATION)

(COMPANY="IBM" AND *stop*(CN_PERvt) < *stop*(ADDRESS_PER vt)

AND BUILDING="Maple House")) CN)

The result of query 6 is shown in Table X.

TABLE X. RESULT OF QUERY 6

| CN |
|---|
| 5.2 |

## VII. BTN-SQL EXTENSION

An extension of SQL, called BTN-SQL, is given in this section using a number of examples in order to demonstrate how the BTNM algebra can be applied and expressed in SQL.

The temporal extension of SQL must satisfy some minimum requirements:

- It must be a simple extension of standard SQL.
- No special treatment must be needed for nested data.
- Queries must be easily written, read and understood.
- Conditions on temporal data must be treated differently than other data; therefore they must be included in another clause than the *Where* clause.
- Valid time and transaction time must be treated similarly (they are only distinguished semantically).
- Nesting levels of attributes must be easily distinguished.
- An attribute must be specified by its name and its attribute path.
- Temporal functions defined in previous temporal models such as CONTAINS and OVERLAPS, are also included in BTN-SQL.

The basic BTN-SQL syntax is given below.

**SELECT <attribute list>**
**FROM <relation list>**
**[WHERE <boolean expression>]**
**[WHEN <temporal expression>]**

### A. Support of subrelations

One of the hardest tasks to be implemented in SQL is the support of nested attributes, i.e. subrelations.

In SQL:2003 the MULTISET data type is introduced. A multiset is an unordered collection of elements, of the same data type [6]. In the present work, MULTISET data type is extended to support the creation and manipulation of nested attributes in general. *EXMULTISET*, the extended MULTISET data type, is an unordered collection of attributes not necessarily of the same data type where duplication is not permitted at the same level.

The CREATE TABLE syntax of BTN-SQL is given below.

**CREATE TABLE <table name> (**
**<attribute_name1> <data type>,**
**<attribute_name2>**
**EXMULTISET (<attribute_name3> <data type>,**
**<attribute_name4> <data type>),**
**… )**

The data type of an attribute in the MULTISET data type could also be another collection type; therefore, multiple nesting levels can be supported.

**Example 2.**

```
CREATE TABLE A (
    X EXMULTISET (X1 CHAR(5),
                  X2 CHAR(5)),
    Y CHAR(3));
```

An instance of nested relation A is given in Table XI.

TABLE XI. NESTED RELATION A INSTANCE

| X | | Y |
|---|---|---|
| X_A | X_B | |
| x_a1 | x_b2 | y1 |
| x_a1 | x_b3 | |
| x_a2 | x_b3 | y2 |

The manipulation of nested relations in different DBMSs has not been resolved efficiently yet. For example, in Oracle10g the ability to support nested relations is given by the keyword *type* which creates a nested table of objects of this type. The keyword *table* allows next the treatment of a nested relation as a normal relation. A new type that is a bag of that type may be created by using ***as table of***. The approach used in Oracle10g is not convenient, manageable and user friendly, given that the number of nesting levels that a nested attribute can have, may be large.


DOI: 01.IJIT.3.3.1249

The insertion of the first tuple in nested relation A (Table 11) is given by the following example.

*Example 3.*

INSERT INTO A values
( ((‘x_a1’, ‘x_b2’), (‘x_a1’, ‘x_b3’)), ‘y1’);

The use of brackets denotes a set a values, so in the first tuple of relation A a set of values of nested attribute X corresponds to value y1 of atomic attribute Y.

BTN-SQL can be applied to subrelations using attribute paths for defining column names, i.e. attribute COURSE of relation BiT_TRAINING is referred as BiT_TRAINING.TRAINER.COURSE; therefore, the nested feature of the model is easily expressed in BTN-SQL. Columns can be reached recursively, regardless the nesting level they are located.

The nesting level of a column is specified by the number of dots its attribute path contains. Therefore, nested attribute Y of DEPT table is specified by its attribute path, DEPT.UNIT.COURSE_DETAILS.C.Y and the nesting level of it can be easily found by counting the number of dots it contains, i.e. 4.

A new function is defined for the computation of the nesting level of an attribute, called NL. NL returns a positive integer. NL is used also to compare the nesting levels of two attributes belonging at two different relations when a join operation is performed. NL is equal to 1 for attributes at the top level of a relation.

*Example 4.*

NL(DEPT.UNIT.COURSE_DETAILS.C.Y)=4

Therefore, the syntax of the SELECT clause in SQL is similar to standard SQL with the extension of using path names instead of plain column names.

The first two queries presented in Section VI using BTN algebra are written below in BTN-SQL.

*Query 1 (BTN-SQL).*

SELECT BTT.COMPANY, BTT.TRAINER.COURSE
FROM BiT_TRAINING BTT

*Query 2 (BTN-SQL).*

SELECT BTT.TRN, BTC.TITLE
FROM BiT_TRAINING BTT, BiT_COURSE BTC
WHERE BTC.COURSE= BTT.TRAINER.COURSE

### *B. Support of time*

In addition to PERIOD data type of SQL3, a new data type, named TELEMENT, as an extension of PERIOD data type, is introduced in BTN-SQL. The TELEMENT data type is used to support temporal elements in BTN-SQL. A temporal element is defined as the union of a set of time intervals (see Section II).

Two temporal elements can be compared by comparing each time interval of the first temporal element by each time interval of the second temporal element and merge the result.

Temporal operators for temporal elements like *start, stop, overlaps, duration, meets, before, after* must be also defined. Some of them are already provided in SQL (see Section VIII).

Queries 3-6, presented in Section VI, are expressed below in BTN-SQL.

*Query 3 (BTN-SQL).*

SELECT *
FROM BiT_TRAINING BTT
WHERE BTT.TRAINER.TRN = ‘Mark’
OR BTT.TRAINER.TRN = ‘Tim’
WHEN BTT.TRAINER.COURSE.CN_PER$_{vt}$ *overlaps* [1/1/2007, 1/1/2008)

*Query 4 (BTN-SQL).*

SELECT BTT.TRN,
*start*(BTT.TRAINER.COURSE.CN_PERT$_{vt}$)
FROM BiT_TRAINING BTT

*Query 5 (BTN-SQL).*

SELECT BTC.TITLE
FROM BiT_TRAINING BTT, BiT_COURSE BTC
WHERE BTT.TRAINER.COURSE=BTC.COURSE
AND BTT.TRAINER.TRN=’Mark’
AND BTC.COURSE.CN_PER$_{It}$ CONTAINS [31/12/2009, 31/12/2009)

*Query 6 (BTN-SQL).*

SELECT BTT.TRAINER.COURSE.CN
FROM BiT_TRAINING BTT, BiT_LOCATION BTL
WHERE BTT.COMPANY=’Ibm’
AND BTL.ANNEX.BUILDING=’Maple House’
WHEN *stop*(BTT.TRAINER.COURSE.CN_PER$_{vt}$) < *stop*(BTL.ANNEX.ADDRESS_PER$_{vt}$)
AND BTT.COMPANY=BTL.COMPANY

From the above examples it is clear that queries can easily be expressed in BTN-SQL, since it is designed to reflect the natural language.

## VIII. Related Work

TSQL2 [7] is the first official effort for the development of a temporal query language. It has been designed as a consistent extension of SQL-92. TSQL2 supports both valid and transaction times. A new data type named PERIOD is introduced. Its range and precision can be expressed as an integer or as an interval. Valid-time tables can be expressed by event or state tables. State tables have tuples timestamped by temporal elements which are unions of periods. In an event table each tuple is timestamped with an instant set. The timestamp is implicit associated with a tuple. Transaction-time tables are implicitly timestamped by temporal elements, one for each tuple that specify when that tuple was logically stored in the database. Thus, an unnamed hidden column associated with the rows of a temporal table was used for expressing period information. In addition, table definition syntax was extended by including the special keywords, AS TRANSACTION TIME and AS VALIDTIME, which were used for creating transaction-time table and valid-time table respectively. In [8] an overview and analysis of TSQL2 is discussed, and major flaws are presented with most significant the existence of ‘hidden attributes’ which makes TSQL2 not relational.

Some parts of TSQL2 were included in a substandard of SQL3 which is the standard SQL:1999. It was called SQL/Temporal [9]. However, it was withdrawn near the end of


DOI: 01.IJIT.3.3.1249

2001. SQL3 permits one or two temporal dimensions to be added to an atemporal relation; therefore, queries run over temporal relations. Techniques for designing and building database applications using SQL are presented in [10].

A valid-time extension of SQL based on the efficient investigation of an Object-Relational database system is proposed in [11]. The approach is a minimal extension of SQL:1999. It maps temporal external queries and data model into an equivalent internal representation. Initial external queries are point-based while resulted internal ones are based on time intervals.

A prototyping tool is presented in [12], called ProSQL, to support the development of extensions to SQL. ProSQL is implemented in the context of temporal extensions to SQL. Generic interval data types are used for the execution of queries using attribute time stamping, valid time state tables and schemas.

A review is given in [13] where temporal aggregation in SQL based temporal query languages on tuple timestamped valid-time data models is supported.

The temporal query language, T4SQL, is proposed in [14] where two new temporal dimensions are introduced, availability time and event time, in addition to valid time and transaction time. The main features of T4SQL are the support of different semantics, *current*, *sequenced*, *atemporal* and *next* and the support of temporal grouping. T4SQL uses constants and standard temporal data types from SQL92 and the PERIOD data type from SQL3.

The choice of using either intervals or temporal elements for timestamping events or objects respectively is investigated in [15]. The interval-based data model uses the tuple-timestamping approach while the temporal element-based data model uses the attribute-timestamping approach. Two temporal query languages are introduced named ISQL and ParaSQL, Interval-based and Parametric Structured Query Languages correspondingly. A query suite has been developed for comparing and evaluating the two temporal data models. The results proved that the interval-based data model shows an increased query complexity due to increased use of self-join operations. However, if self-joins are treated by a special algorithm, the two models have similar performance.

In [16] the Nested Bitemporal Relational Model is implemented by bitemporal atoms that need new abstract data types to be defined. Bitemporal atoms use built-in data type DATE for the lower and upper bounds of transaction and valid times. Since a bitemporal atom contains transaction time lower and upper bounds, valid time lower and upper bounds and a value, the SUBSTR function needs to be used in most of the queries.

SQL:2011 [1], the latest edition of the SQL standard, supports temporal features and specifically, the creation and manipulation of temporal tables. The main contribution of SQL:2011 concerning temporal data support is the *period definition* as metadata to tables identifying a pair of columns capturing the period start and end times. These columns must be of either DATE or a timestamp type with same data types. An application-time period table supports valid time and a system-versioned table supports transaction time. A table may be both an application-time period table and a system-versioned table. New features of SQL:2011 include the ability to specify changes effective within a specified period. The following period predicates are provided, CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES and IMMEDIATELY SUCCEEDS.

To conclude, many proposals and extensions of SQL have appeared in the literature on the domain of bitemporal databases over the years, however work still needs to be done. One of the ongoing problems that remains is the way bitemporal nested relations are treated and this is the subject of the current research work.

## IX. Conclusions

Although intensive research has been undertaken for many years in the field of bitemporal databases, a number of issues require further investigation. One of these issues is the extension of SQL standard for the support of bitemporal nested relations. This paper represents one attempt towards this direction. Additionally, and for reasons of facilitation, the modelling of complex nested data has been revised and a new approach is proposed, the BTN-ER model.

An implementation is under development. Future work includes the study of optimization techniques for the efficient evaluation of complex queries.

## References

[1] K. Kulkarni, and J. E. Michels, "Temporal Features in SQL:2011," SIGMOD Record, vol. 41, no. 3, pp. 34-43, September 2012.

[2] G. Garani, "An Algebra for the BiTemporal Nested Data Model," Proceedings of the 3rd International Conference on Advances in Information and Communication Technologies, Amsterdam, Netherlands, 22-23 November, 2012.

[3] G. Garani, "Recursive Natural Join Operation in Bitemporal Nested Relations," Intern. J. Intell. Inf. and Database Syst., vol. 7, no. 4, pp. 356-372, 2013.

[4] A. U. Tansel, and C. E. Atay, "Nested Bitemporal Relational Algebra," Proceedings of ISCIS 06, Istanbul, Turkey, 1-3 November, Springer-Verlag, Berlin, pp. 622-633, 2006.

[5] A. U. Tansel, "Temporal Relational Data Model," IEEE Trans. on Knowl. and Data Eng., vol. 9, pp. 464-479, June 1997.

[6] A. Eisenberg, J. Melton, K. Kulkarni, J. E. Michels, and F. Zemke, "SQL:2003 Has Been Published," SIGMOD Record, vol. 33, no. 1, pp. 119-126, March 2004.

[7] R. Snodgrass (Ed.), The TSQL2 Temporal Query Language, Kluwer Academic Publishers, 1995.

[8] H. Darwen, C. J. Date, "An overview and Analysis of Proposals Based on the TSQL2 Approach," In Date on Database: Writings 2000-2006, C. J. Date, Apress, pp. 481-514, 2006.

[9] R. Snodgrass, M. H. Böhlen, C. S. Jensen, and A. Steiner, "Transitioning temporal support in TSQL2 to SQL3," A TIMECENTER Technical Report, TR-8, 1997.

[10] R. Snodgrass, Developing Time-Oriented Database Applications in SQL, Morgan Kaufmann, 1999.


DOI: 01.IJIT.3.3.1249

[11] C. X. Chen, J. Kong, and C. Zaniolo, "Design and Implementation of a Temporal Extension of SQL," Proceedings of the 19th International Conference on Data Engineering, 5-8 March, pp. 689-691, 2003.

[12] J. Green, and R. Johnson, "ProSQL: A Prototyping Tool for SQL Temporal Language Extension," Proceedings of the 20th British National Conference on Databases, Coventry, UK, 15–17 July, New Horizons in Information Management. Lecture Notes in Computer Science, Volume 2712, pp.190-197, 2003.

[13] M. H. Böhlen, J. Gamper, and C. S. Jensen, "How would you like to aggregate your temporal data?," 13th International Symposium on Temporal Representation and Reasoning, 15-17 June, pp. 121–136, 2006.

[14] C. Combi, A. Montanari, and G. Pozzi, "The T4SQL Temporal Query Language," Proceedings of the 16th ACM Conference on Information and Knowledge Management, Lisboa, Portugal, 6-8 November, pp. 193-202, 2007.

[15] S.-Y. Noh, and S. K. Gadia, "Benchmarking temporal database models with interval-based and temporal element-based timestamping," The J. of Syst. and Soft., vol. 81, pp. 1931-1943, November 2008.

[16] C. E. Atay, and A. U. Tansel, Bitemporal Databases: Modeling and Implementation, VDM Verlag Dr. Müller, 2009.


DOI: 01.IJIT.3.3.1249